02

Métodos tradicionales usando Collections y Streams

Transcripción

[00:00] Hola. ¿Cómo están? Vamos a iniciar nuestra clase número 9, que está dentro del bloque Conociendo más de lista. Ahora vamos a ver un poco de métodos tradicionales usando collections, y también vamos a introducir más la parte de stream, que viene a partir de Java 8, cómo utilizar lambdas.

[00:21] Para esto primero la clase 8 vamos a cambiarla para clase 9, pero nuestra clase 8 fue un poco diferente, entonces vamos a copiar todo lo de la clase 5 que estuvimos trabajando y vamos a hace aquí en lugar de curso vamos a colocar curso de historia, de álgebra, de aritmética, de geometría.

[01:04] Adicionamos los cursos y aquí tenemos por ejemplo Collections.sort y ordenamos por nombre e imprimimos nuevamente. Pero ahora, supongamos que nosotros queramos ver el tiempo total de todos nuestros cursos. Para esto tendremos que hacer un for, colocamos aquí curso y después hacemos por ejemplo cursos.

[01:36] Una vez hecho esto, vamos a colocar aquí una variable tiempo = 0, y aquí vamos a hacer tiempo es igual, y vamos a estar sumando, que sería curso.getTiempo. Ya no vamos a imprimir listas, por ejemplo, que eso ya lo vimos anteriormente. Y ahora vamos a ver aquí System.out.println(tiempo).

[02:18] Vamos a ver aquí, 110 tenemos de tiempo. Imaginémonos que nosotros queramos hacer esto con streams. ¿Qué sucedería? Vamos a hacer aquí lo siguiente. Para streams sería una forma bien simple. Tenemos aquí

cursos.stream().mapToInt, decimos que sería aquí la clase. Estamos con curso, disculpen, estamos con curso. Y getTiempo, que es entero.

[03:13] Una vez hecho esto, utilizamos aquí sum. ¿Qué estamos haciendo? Estamos haciendo un stream, estamos haciendo un map, juntando todo el entero y estamos obteniendo getTiempo y estamos sumando todos los resultados de getTiempo. Para esto listamos y tenemos la misma respuesta, 110, 110.

[03:46] Aquí sería 70, 80, 110. Prácticamente tuvo la misma respuesta. Solamente que aquí tuvimos que crear una variable, tuvimos que crear un for, y con stream se nos hizo bien más simple. Por ejemplo, queremos el número máximo, colocamos .max y ejecutamos nuevamente.

[04:12] Aquí en el max, él devuelve aquí que está diciendo un optional, pero aquí ya me dijeron que el max es 50, pero yo quiero que aquí me devuelva verdaderamente un número, entonces quiero aquí que sea getAsInt, quiero que me devuelva verdaderamente el número mayor, que me lo convierta, 50.

[04:38] Todo prácticamente con stream está orientado más con métodos, entonces se hace bien más simple hacer cosas que anteriormente eran más complicadoas, por ejemplo, aquí para encontrar el número mayor, primero tendríamos que aquí ya colocar esta variable llamada mayor y ver aquí con un if, simple.

[05:06] Si getTiempo es mayor que número mayor, ¿qué hacemos? Mayor es igual a curso.getTiempo. Y aquí imprimo el número mayor, y aquí ejecutamos. 50 también retornó aquí. Aquí lo quitamos aquí de nuevo, utilizo el mapToInt.sum. No, .max. y .getAsInt. 50 y 50.

[06:06] Facilidad, aquí tenemos prácticamente una línea y aquí tuvimos que crear una variable más. Entonces, tenemos muchas opciones para nosotros poder usar stream, tenemos varios tipos de utilización para stream, lo que se nos va a facilitar un poco trabajar con collection. Es bueno también recordar

que también stream puede bajar un poco el performance de una lista, pero también puede mejorar bastante cuando tenemos una lista más moderada.

[06:38] Todo es cuestión de probar y ver tiempos, que ese es un punto bien importante. Pero el objetivo es ahora ver que todas las opciones que podemos tener utilizando stream. Por ejemplo ahora queremos que también él sume, último ejemplo, vamos a sumar toda la lista, pero primero vamos a hacer un filter y ya vamos a tener que lo mismo que hicimos aquí, que no tenga curso y aquí que no tenga historia.

[07:32] Queda como resultado 80. Le quité historia. ¿Historia cuánto era? 30. Esto sería 110 - 30 = 80. Sumo aquí, 10 + 20 + 50. ¿Qué fue? Solamente coloqué aquí un filter y dije: "No quiero que tenga historia". En el caso de aquí, por ejemplo, si bien aquí ahora sería aquí suma, tendría que hacer lo siguiente: += curso.getTiempo().

[08:08] Y una vez hecho esto, tenemos que ver primero si el curso.getNombre igual, aquí, equalsIgnoreCase y aquí. Vamos a utilizarlo de aquí. Aquí estamos creando, ¿qué cosa? Un if. Y aquí suma. 80 y 80. Entonces prácticamente tuvimos el mismo resultado, solamente que aquí utilizando collections o no utilizando stream tuvimos otro tipo de dificultades, más código, y con este if termina siendo un código un poco más limpio.

[09:00] No digo que stream sea lo mejor pero para ciertas circunstancias es muy bueno trabajar con él. Tenemos otros métodos que vamos a ver en nuestra próxima clase, métodos adicionales, tipo filter, tipo map, tipo sort, ya profundizar más esos métodos utilizando stream, esto sería todo en nuestra clase número 9. Nos vemos en la siguiente, muchas gracias.